Chap5: Embedding, Mapping, Scheduling

Figures from the book

Program Graph $\rightarrow$ Processor Network

LOAD
DILATION
CONGESTION
Figure 5-1

Embedding

(a) Dilatation = 2
    Load = 1

(b) Congestion = 2
    Load = 1
Figure 5-2

$G$

$G'$

load = dilatation = congestion
Figure 5-3

\[ G \]
\[ G' \]

DILATION = ?
CONGESTION = ?
RING INTO 2-D MESH

R_{20} INTO 4x5 MESH

R_{25} INTO 5x5 mesh?
2D mesh into 2-D meshes

side way or column wise fit.
Binary Tree into 2D mesh

Depth 3 binary tree

Congestion-free dilation-1 embedding
Theorem: A complete binary tree of depth $k > 4$ has no $dilation - 1$ embedding on a 2D mesh.

Proof: 2D mesh

Number of nodes with distance $k$

$= 1 + 4 + 2 \times 4 + \ldots + 4(k-1) + 4k$
$= 1 + 4(1 + 2 + 3 + \ldots + k)$
$= 1 + 4 \frac{k(k+1)}{2}$
$= 1 + 2k(k+1)$

Number of nodes in a 2D mesh with depth $k$

$= 2^0 + 2^1 + \ldots + 2^k$
$= 2^{k+1} - 1$

H.W. Prove, by induction, that
Figure 5-7

H-Tree based embedding

DILATION = 2
Binomial Tree $\rightarrow$ 2D Mesh

Figure 5-8
DILATION $\frac{F_k}{27}$ for depth $k$ tree
Embedding Binomial Trees onto Hypercubes

- depth 4 Binomial Tree onto dimension 4 HC
Ring $\rightarrow$ HC

$R_8$ onto HC of dimension 3

2D Mesh to HC

4x4 mesh \rightarrow 2^4 \text{ node HC}
Reflected Binary Codes (Gray Codes)

0   0   0   0   0
1   0   1   0   0
   1   1   0   0
   1   0   0   0

0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0

00  00  01  11  10
01  10  11  10  01
11  11  10  01  00
10  01  00  11  10

00  01  11  10  10
01  10  01  00  11
11  11  10  01  00
10  01  00  11  10

2  3  0  1  0
6  7  1  1  1
Figure 5-15: Static Scheduling of Dependency Graphs.
Dynamic Load Balancing

- Battlefield Management Simulation
- Numerical Integration

1. Centralized Load Balancing
   - remapping after each phase
   - task migration

2. Distributed Load Balancing
   - load sharing among neighbors
   - global distribution through local deci

3. Semi-Distributed Load Balancing
   - centralized within regions, distributed
Deadlocks

Shared Variables

\( P_0 \)

\( \text{lock}(A) \)

\( \text{lock}(B) \)

Distributed Programs

\( P_0 \)

get \( x \) from \( P_1 \)

send \( y \) to \( P_1 \)

\( P_1 \)

get \( y \) from \( P_0 \)

send \( x \) to \( P_0 \)
Deadlock in buffers

read X from Processor i

write X to Processor 0

Processor 0

Processor i
Causes of Deadlocks

- Mutual Exclusion of shared resources
- non-preemption
- indefinite wait
- cyclic wait

Solutions

1. Detection & Resolution

2. Avoidance
   - Ensure overall state of resource allocation will remain safe as they are allocated

3. Prevention
   - Order Resources